IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Dhruva R. Chakrabarti et al. | Examiner: Wu, Junchun

Serial No. 10/699,144 | Art Unit: 2191

Filing Date: October 31, 2003 | Attorney Docket No.: 200313003-1

Title: Cross-File Inlining by Using Summaries and Global Worklist

---

Honorable Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF FILED UNDER 37 C.F.R. § 41.37

Sir:

This appeal brief is being filed with a Notice of Appeal in response to the latest office action mailed on June 20, 2008.

The Commissioner is hereby authorized to charge requisite fees due for this submission to Deposit Account No. 08-2025 (Hewlett Packard).

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Development Company, L.P., a Texas Limited Partnership having its principal place of business in Houston, Texas. The Hewlett-Packard Development Company, L.P., is the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

On information and belief, there are no appeals, interferences, or judicial proceedings known to the appellant, the appellant's legal representative, or assignee which may be related to, directly affect or be directly affected by or have a bearing on the Board of Patent Appeals and Interferences (the "Board") decision in the pending appeal.

III. STATUS OF CLAIMS

A. Total Claims: 1-21

B. Current Status of Claims:

1. Claims canceled: 2 and 9
2. Claims withdrawn: none
3. Claims pending: 1, 3-8 and 10-21
4. Claims allowed: none
5. Claims rejected: 1, 3-8 and 10-21
6. Claims objected to: none

C. Claims on Appeal: 1, 3-8 and 10-21

As indicated above, claims 1, 3-8 and 10-21 are pending in this application, stand finally rejected, and are being appealed. These claims are rejected in the final office action mailed June 20, 2008 ("the last office action").

IV. STATUS OF AMENDMENTS

No amendment has been filed after the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed subject matter relates to computer software. More particularly, the claimed subject matter relates to software compilers.

Independent claim 1 relates to a method of compiling a computer program (page 2, lines 21-22). A plurality of modules of source code are received (f1.c, f2.c, ..., fn.c in Fig. 1A and description on page 6, lines 27-29). Intermediate representations corresponding to the modules are generated (f1.0, f2.o, ..., fn.o which are output by IPO in Fig. 1A and description on page 6, lines 32-33). A set of data from the intermediate representations is extracted to create an inliner summary for each module (block 408 in Fig. 4 and description on page 11, lines 17-26). The inliner summaries (page 11, line 15 through page 16, line 26) and a globally-sorted working-list based order (page 16, line 30 through page 17, line 4) are used in an inline analysis phase (FIG. 5 and page 19, line 16 through page 20, line 29), without using the intermediate representations in the inline analysis phase (page 3, lines 11-13 and 20-22; page 21, lines 9-11), to determine which call sites in the modules are to be inlined by substituting code from a called module (page 2, lines 26-27). The globally-sorted working-list based order is dynamically updated during the inline analysis phase (page 20, lines 17-29). After a call site is determined to be inlined, a call graph of the routines and call sites is updated, and the inliner summaries throughout the call graph are updated (page 3, lines 13-14; page 20, line 17 through page 21, line 11). Determining the call sites to be inlined involves proceeding only once through the call sites in said dynamically-updated globally-sorted working-list based order (FIG. 5 and page 28, lines 24-28).

Independent claim 8 relates to an apparatus for compiling a computer program (page 2, lines 28-29). The apparatus includes a processor configured to execute computer-readable code, a memory system configured to store data, computer-readable code for a front-end portion of a compiler program and for a cross-module optimizer of the compiler program (page 2, lines 29-30). The front-end portion of the compiler program is configured to receive a plurality of modules of source code, generate intermediate representations corresponding to the modules, and extract a set of data from the intermediate representations to generate inliner summaries for the modules (page 2, lines 30-33). The cross-module optimizer is configured to use the inliner summaries and

a dynamically-updated globally-sorted working-list based order to analyze the call sites in an inline analysis phase, without using the intermediate representations (page 3, lines 11-13 and 20-22; page 21, lines 9-11), so as to determine which call sites in the modules are to be inlined by substituting code from a called module (page 2, line 33 through page 3, line 4). The cross-module optimizer is further configured to update a call graph of the routines and call sites (page 20, lines 17-29), and to update the inliner summaries (page 3, lines 13-14; page 20, line 17 through page 21, line 11), after a call site is determined to be inlined. Determining the call sites to be inlined involves proceeding only once through the call sites in said dynamically-updated globally-sorted working-list based order (FIG. 5 and page 28, lines 24-28).

Independent claim 15 relates to a computer program product comprising a computer-usable medium having computer-readable code embodied therein. The computer program product is compiled from a plurality of modules of source code using inliner summaries and a dynamically-updated globally-sorted working-list based order (page 16, line 30 through page 17, line 4; page 20, lines 17-29) in an inline analysis phase, without using intermediate representations (page 3, lines 11-13 and 20-22; page 21, lines 9-11), to determine which call sites in the modules are to be inlined by substituting code from a called module (page 2, line 33 through page 3, line 4) and, after a call site is determined to be inlined, to determine a call graph of the routines and call sites, and to update the inliner summaries throughout the call graph (page 3, lines 13-14; page 20, line 17 through page 21, line 11). Determining the call sites to be inlined involves proceeding only once through the call sites in said dynamically-updated globally-sorted working-list based order (FIG. 5 and page 28, lines 24-28).

Independent claim 16 relates to a method of compiling a computer program with a plurality of modules of source code and corresponding intermediate representations (page 2, lines 21-23). A set of data is extracted from the intermediate representations of the modules to create an inliner summary for each module (page 2, lines 23-25). The inliner summaries are used in a one-pass inline analysis phase (FIG. 5 and page 28, lines 24-28), without using the intermediate representations (page 3, lines 11-13 and 20-22; page 21, lines 9-11), to determine which call sites to inline (page 2, lines 26-27), in what order to inline them (page 20, lines 17-29; page 23, lines 29-32), and preserving a same order

during the transformation phase (page 29, lines 11-13). A measure of goodness for each call site is formulated from the inliner summary (page 3, lines 16-20). A technique is used to dynamically update information in the summary for potentially all call-graph nodes and edges every time a call site is accepted for inlining (page 3, lines 13-14; page 20, line 17 through page 21, line 11). A globally sorted work-list (page 33, lines 1-2 and lines 8-10) and an associated table (page 32, lines 29-33) are used to maintain and manipulate the call sites. The work-list is dynamically updated every time a call site is accepted for inlining (page 20, lines 17-29).

Applicants respectfully submit that independent claims 1, 8, 15 and 16 do not include any means-plus-function or step-plus-function elements under the sixth paragraph of 35 U.S.C. § 112.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following grounds of rejection are to be reviewed on appeal:

1. The rejection of claims 1, 4, 5, 7, 8, 11, 12 and 14-21 under 35 U.S.C. §103 (a) as being unpatentable over Ayers et al. ("Aggressive Inlining," 1997, ACM) in view of Schmidt (US Patent No. 6,195,793); and

2. The rejection of claims 3, 6, 10 and 13 under 35 U.S.C. §103 (a) as also being unpatentable over Ayers et al. ("Aggressive Inlining," 1997, ACM) in view of Schmidt (US Patent No. 6,195,793).

VII. ARGUMENT

Applicants respectfully traverse the aforementioned rejections of claims 1, 3-8 and 10-21 in the latest office action for the following reasons.

A. Claims 1 and 3-7 overcome their rejections under 35 U.S.C. § 103(a)

Claims 1 and 3-7 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Ayers et al. ("Aggressive Inlining," 1997, ACM) in view of Schmidt (US Patent No. 6,195,793). Applicants respectfully traverse this rejection.

Claim 1 recites as follows.

> 1. A method of compiling a computer program, the method comprising:
> receiving a plurality of modules of source code;
> generating intermediate representations corresponding to the modules;
> extracting a set of data from the intermediate representations to create an inliner summary for each module;
> using the inliner summaries and a globally-sorted working-list based order in an inline analysis phase, without using the intermediate representations in the inline analysis phase, to determine which call sites in the modules are to be inlined by substituting code from a called module, wherein **said globally-sorted working-list based order is <u>dynamically updated</u>** during the inline analysis phase; and
> after a call site is determined to be inlined, updating a call graph of the routines and call sites, and updating the inliner summaries throughout the call graph,
> wherein determining the call sites to be inlined involves **<u>proceeding only once</u> through the call sites in said dynamically-updated globally-sorted working-list based order.**

(Emphasis added.)

(i) NEITHER AYERS ET AL. NOR SCHMIDT TEACH **DYNAMICALLY UPDATING THE ORDER OF CALL SITES IN THE WORKING LIST**

Applicants respectfully submit that the claim limitation that "**said globally-sorted working-list based order is dynamically updated** during the inline analysis phase"

(emphasis added) is contrary to the disclosure and teachings of both Ayers et al. and Schmidt.

This claim limitation is discussed, for example, in the original application at pages 20 and 23, for example. Page 20, lines 17-29 recites as follows.

> **If the call site is to be inlined, then the process continues on to update 510 the call graph, the routine summaries, and the working list**. The summaries may need to be updated by iterating on the call-graph. This is necessary in order to capture the effects of inlining a certain call site on various routines. This is emphasized by the illustration of a call-graph shown in FIG. 8. The dashed call-graph edge (also numbered 1) denotes the call site that has been accepted for inlining. The dotted dashed lines indicate how the summaries will need to be propagated through the call-graph.
>
> In accordance with one embodiment, new edges may need to be created after inlining a call site. When a caller routine A inlines a callee routine B, for every edge from routine B to a routine C, a new edge needs to be created from routine A to routine C. Each of these new edges is a candidate for further inlining and hence is inserted in the working list.

(Emphasis added.)

In addition, page 23, lines 29-32 recites as follows.

> If the caller or the callee routine for a certain call site has its summary information modified, the goodness factor of that call site is recomputed. **Whenever the goodness factor of a certain call site is recomputed, the old entry is deleted from the working list and a new updated entry is inserted.**
>
> (Emphasis added.)

In other words, the working list for the inline analysis phase is dynamically updated based on changes in the goodness factor of call sites.

In contrast, Ayers et al. teaches using statically-sorted priority lists throughout its procedure to perform the inline analysis. This is supported by the following citation. For example, section 2.4, third paragraph, lines 1-4 of Ayers et al., states as follows. "The inliner then walks over the inline site list **in priority order**. The compile-time impact of each site is considered, and if within the current budget, the inline is accepted."

Hence, Ayers teaches going through the inline site list in a static "priority" order. There is no teaching or suggestion in Ayers et al. of the claimed feature of using a **dynamically updated** working-list based order. The latest office action agrees with this deficiency of Ayers et al. (see page 3 of the office action).

Applicants respectfully submit that Schmidt also teaches using statically-sorted priority lists throughout its procedure to perform the inline analysis.

Regarding the first approximation phase, Schmidt teaches using an initial priority queue which is statically calculated to go through the call sites (called "arcs" in Schmidt) and then discarding that initial priority queue. This is clearly stated in column 5, lines 44-65, which recites as follows. "**In the first approximation phase of FIG. 2, all inline candidates or arcs with finite priority are placed in an initial priority queue.** Then as indicated at a block 306, the best inline candidates are removed from the initial priority queue, marking them as tentatively to be inlined, until the code bloat budget has been exhausted." (Emphasis added.) Therefore, Schmidt teaches going through the call sites according to a static "initial priority queue" in the first approximation phase.

Regarding the refinement stage, Schmidt teaches the use of two static queues, a "ReadyQueue" and an "AuxQueue." The disclosure by Schmidt in regards to these two queues is discussed below.

As stated in col. 6, lines 5-13, "A ReadyQueue contains all procedures that can be issued next in the compilation order, either because they are leaf procedures or because all of their children have already been placed in the compilation order. The procedures in the ReadyQueue are sorted based on the inlining priority of the best incoming arc for each procedure. That is, given that either of two procedures could be issued at a given time, the one we choose should be the one that has the highest inlining value, because it is called from a high priority call site." Hence, **Schmidt teaches that the ReadyQueue is a queue that is sorted based on priority.** There is <u>no</u> teaching or suggestion in Schmidt that the ReadyQueue has an order which is dynamically updated.

As stated at col. 6, lines 17-20, "The AuxQueue contains initially outlined call sites whose final bloat is known, and that may be used to replace those candidates that have bloated unacceptably since they were initially selected." Hence, **Schmidt teaches that the AuxQueue is a queue containing initially outlined call sites whose final bloat**

is known. There is <u>no</u> teaching or suggestion in Schmidt that the AuxQueue has an order which is dynamically updated.

Thus, applicants respectfully submit that there is <u>no</u> disclosure in Schmidt as to the claimed **working-list based order which is dynamically updated.**

Hence, for at least this reason, applicants respectfully submit that amended claim 1 overcomes its rejection.

(ii) NEITHER AYERS ET AL. NOR SCHMIDT TEACH OR SUGGEST **PROCEEDING ONLY ONCE** THROUGH THE WORKING LIST OF CALL SITES

Claim 1 recites that "determining the call sites to be inlined involves **proceeding only once** through the call sites in said dynamically-updated globally-sorted working-list based order." (Emphasis added) This claim limitation is supported, for example, in the original application by the flow chart of FIG. 5 and the description thereof. As shown in FIG. 5, the process proceeds only once through the working list of call sites. As described on page 28, lines 24-28 (emphasis added), "In addition to updating **510** the call graph and inliner summaries, dependence information is also updated **512** in accordance with an embodiment of the invention. Thereafter**, the determination is made 514 as to whether there are any more call sites on the working list to be analyzed. If so, then the process loops back to selecting 502 the most profitable call site remaining, followed by the legality analysis 504, and so on. If not, then the inline analysis phase 304 terminates** and the inline transformation phase **306** is entered."

In contrast, Ayers et al. teaches a multi-pass inliner which performs an inline analysis multiple times. See, for example, Ayers et al., page 135, right column, line 5, "HLO performs **several passes** of inlining and cloning." (Emphasis added.) The latest office action agrees with this deficiency of Ayers et al. (see page 3 of the office action).

Applicants respectfully submit that Schmidt also proceeds through the call sites in **more than one pass**. As discussed in Schmidt, the **initial pass** uses the initial priority queue. In particular, col. 5, lines 59-67 of Schmidt states as follows.

> **In the first approximation phase of FIG. 2, all inline candidates or arcs with finite priority are placed in the initial priority queue.** Then as indicated at a block 306, **the best inline candidates are removed from the initial priority queue, marking them as tentatively to be inlined, until the code bloat budget has been exhausted. This initial queue is then discarded. At this point the best sites to be inlined have been determined**, provided that none of the inlined procedures grow because they themselves contain inlined call sites.

(Emphasis added.)

After the initial pass, the technique taught by Schmidt makes a **subsequent pass** in a refinement stage. In this subsequent pass, Schmidt's technique proceeds from the leaves towards the root of the call-graph and compares the profit of the already-accepted call-sites with the profit from the elements in AuxQueue. The elements in AuxQueue are the not-already-inlined call sites that were encountered during this upward traversal of the call graph. Depending on the comparison, the subsequent pass taught by Schmidt refines the decisions made in the initial pass. In particular, col. 6, lines 1-20 of Schmidt states as follows.

> **In the refinement stage**, the compilation order for procedures is determined and **the determination of which procedures to inline possibly is changed, based on changes in procedure size**. Two more priority queues are used to accomplish this. A ReadyQueue contains all procedures that can be issued next in the compilation order, either because they are leaf procedures or because all of their children have already been placed in the compilation order. The procedures in the ReadyQueue are sorted based on the inlining priority of the best incoming arc for each procedure. That is, given that either of two procedures could be issued at a given time, the one we choose should be the one that has the highest inlining value, because it is called from a high priority call site. The intent of this is not only to process important call sites as soon as possible, but also to

populate the AuxQueue with high priority call sites that did not make the initial cut of inline candidates. The AuxQueue contains initially outlined call sites whose final bloat is known, and that may be used to replace those candidates that have bloated unacceptably since they were initially selected.

Thus, Schmidt clearly teaches proceeding through the call sites **in more than one pass**, while claim 1 recites "**only once**".

Therefore, claim 1 overcomes its rejection for one or both of the above-discussed reasons.

Claims 3-7 depend from claim 1. Hence, claims 3-7 overcome their rejections for at least the reasons discussed above in relation to claim 1.

B. Claims 8 and 10-14 overcome their rejections under 35 U.S.C. § 103(a)

Claims 8 and 10-14 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Ayers et al. ("Aggressive Inlining," 1997, ACM) in view of Schmidt (US Patent No. 6,195,793). Applicants respectfully traverse this rejection.

Similar to claim 1, claim 8 recites "the cross-module optimizer being configured to use the inliner summaries and a **dynamically-updated globally-sorted working-list based order** to analyze the call sites in an inline analysis phase." Hence, claim 8 overcomes its rejection for at least the reason discussed above in section A (i).

Also similar to claim 1, claim 8 recites "wherein determining the call sites to be inlined involves **proceeding only once through the call sites in said dynamically-updated globally-sorted working-list based order**." Hence, claim 8 further overcomes its rejection for at least the reason discussed above in section A (ii).

Therefore, applicants respectfully submit that claim 8 overcomes its rejection for at least the reasons discussed above in section A in relation to claim 1.

Claims 10-14 depend from claim 8. Hence, claims 10-14 overcome their rejection for at least the reasons discussed above in relation to claim 8.

C. Claim 15 overcomes its rejection under 35 U.S.C. § 103(a)

Claim 15 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over Ayers et al. ("Aggressive Inlining," 1997, ACM) in view of Schmidt (US Patent No. 6,195,793). Applicants respectfully traverse this rejection.

Similar to claim 1, claim 15 recites "a **dynamically-updated globally-sorted working-list-based order** in an inline analysis phase." Hence, claim 15 overcomes its rejection for at least the reason discussed above in section A (i).

Also similar to claim 1, claim 15 recites recites "wherein determining the call sites to be inlined involves **proceeding only once through the call sites in said dynamically-updated globally-sorted working-list based order**." Hence, claim 15 further overcomes its rejection for at least the reason discussed above in section A (ii).

Therefore, applicants respectfully submit that claim 15 overcomes its rejection for at least the reasons discussed above in section A in relation to claim 1.

D. Claims 16-21 overcome their rejections under 35 U.S.C. § 103(a)

Claims 16-21 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Ayers et al. ("Aggressive Inlining," 1997, ACM) in view of Schmidt (US Patent No. 6,195,793). Applicants respectfully traverse this rejection.

Similar to claim 1, claim 16 recites "using a globally sorted work-list and an associated table to maintain and manipulate the call sites and **dynamically updating the work-list** every time a call site is accepted for inlining." Hence, claim 16 overcomes its rejection for similar reasons as discussed above in section A (i).

Furthermore, applicants respectfully submit that claim 16 further specifies that the dynamic updating of the work-list is performed "... **every time a call site is accepted for inlining.**" Applicants respectfully submit that neither Ayers et al. nor Schmidt teaches or suggests this further claim limitation.

Also similar to claim 1, claim 16 recites "**using the inliner summaries in a one-pass inline analysis phase**, without using the intermediate representations, to determine

which call sites to inline, in what order to inline them, and preserving a same order during the transformation phase." Hence, claim 16 overcomes its rejection for similar reasons as discussed above in section A (ii).

Therefore, applicants respectfully submit that claim 16 overcomes its rejection for the above-discussed reasons.

Claims 17-21 depend from claim 16. Hence, claims 17-21 overcome their rejection for at least the reasons discussed above in relation to claim 16.

VIII. CONCLUSION

For at least the above reasons, applicants respectfully request that the rejections of claims 1, 3-8 and 10-21 be overturned.

Respectfully submitted,
DHRUVA R. CHAKRABARTI
ET AL.

Dated: September 8, 2008

/ James K. Okamoto, Reg. No. 40,110/

James K. Okamoto, Reg. No. 40,110
Okamoto & Benedicto LLP
P.O. Box 641330
San Jose, CA 95164
Tel.: (408) 436-2110
Fax.: (408) 436-2114

CERTIFICATE OF MAILING			
I hereby certify that this correspondence, including the enclosures identified herein, is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below. If the Express Mail Mailing Number is filled in below, then this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service pursuant to 37 CFR 1.10.			
Signature:			
Typed or Printed Name:		Dated:	
Express Mail Mailing Number (*optional*):			

CLAIMS APPENDIX

CLAIMS INVOLVED IN THE APPEAL

1. A method of compiling a computer program, the method comprising:
receiving a plurality of modules of source code;
generating intermediate representations corresponding to the modules;
extracting a set of data from the intermediate representations to create an inliner summary for each module;
using the inliner summaries and a globally-sorted working-list based order in an inline analysis phase, without using the intermediate representations in the inline analysis phase, to determine which call sites in the modules are to be inlined by substituting code from a called module, wherein said globally-sorted working-list based order is dynamically updated during the inline analysis phase; and
after a call site is determined to be inlined, updating a call graph of the routines and call sites, and updating the inliner summaries throughout the call graph,
wherein determining the call sites to be inlined involves proceeding only once through the call sites in said dynamically-updated globally-sorted working-list based order

3. The method of claim 1, further comprising, after the call graph and inliner summaries are updated,
re-calculating profitabilities associated with remaining call sites; and
re-ordering the working list using the re-calculated profitabilities.

4. The method of claim 3, wherein updating the inliner summaries comprises determining nodes and edges of the call graph that are affected by the

inlining of the call site and updating those inliner summaries corresponding to the affected nodes and edges.

5. The method of claim 4, wherein the edge summaries include at least a call site execution count and a signature type.

6. The method of claim 4, wherein the node summaries include at least a code size, a routine execution count, and a call-graph height.

7. The method of claim 1, wherein the inline analysis phase is separate and distinct from an inline transformation phase.

8. An apparatus for compiling a computer program, the apparatus comprising:

a processor configured to execute computer-readable code;

a memory system configured to store data;

computer-readable code for a front-end portion of a compiler program, the front-end portion of the compiler program being configured to receive a plurality of modules of source code, generate intermediate representations corresponding to the modules, and extract a set of data from the intermediate representations to generate inliner summaries for the modules; and

computer-readable code for a cross-module optimizer of the compiler program, the cross-module optimizer being configured to use the inliner summaries and a dynamically-updated globally-sorted working-list based order to analyze the call sites in an inline analysis phase, without using the intermediate representations, so as to determine which call sites in the modules are to be inlined by substituting code from a called module, and to update a call graph of the routines and call sites, and to update the inliner summaries, after a call site is determined to be inlined, wherein determining the

call sites to be inlined involves proceeding only once through the call sites in said dynamically-updated globally-sorted working-list based order.

10. The apparatus of claim 8, wherein the cross-module optimizer is further configured to re-calculate profitabilities associated with remaining call sites, and re-order the working list using the re-calculated profitabilities, after the call graph and inliner summaries are updated.

11. The apparatus of claim 10 wherein the cross-module optimizer is further configured to update the inliner summaries by determining nodes and edges of the call graph that are affected by the inlining of the call site and update those inliner summaries corresponding to the affected nodes and edges.

12. The apparatus of claim 11, wherein the edge summaries generated by the front-end portion include at least a call site execution count and a signature type.

13. The apparatus of claim 11, wherein the node summaries generated by the front-end portion include at least a code size, a routine execution count, and a call-graph height.

14. The apparatus of claim 8, wherein the cross-module optimizer is further configured to perform the inline analysis phase separately and distinctly from an inline transformation phase.

15. A computer program product comprising a computer-usable medium having computer-readable code embodied therein, the computer program product being compiled from a plurality of modules of source code using inliner summaries and a dynamically-updated globally-sorted working-list

based order in an inline analysis phase, without using intermediate representations, to determine which call sites in the modules are to be inlined by substituting code from a called module and, after a call site is determined to be inlined, to determine a call graph of the routines and call sites, and to update the inliner summaries throughout the call graph, wherein determining the call sites to be inlined involves proceeding only once through the call sites in said dynamically-updated globally-sorted working-list based order.

16. A method of compiling a computer program with a plurality of modules of source code and corresponding intermediate representations, the method comprising:

extracting a set of data from the intermediate representations of the modules to create an inliner summary for each module;

using the inliner summaries in a one-pass inline analysis phase, without using the intermediate representations, to determine which call sites to inline, in what order to inline them, and preserving a same order during the transformation phase;

formulating a measure of goodness for each call site from the inliner summary;

using a technique to dynamically update information in the summary for potentially all call-graph nodes and edges every time a call site is accepted for inlining; and

using a globally sorted work-list and an associated table to maintain and manipulate the call sites and dynamically updating the work-list every time a call site is accepted for inlining.

17. The method of claim 16, wherein the inliner summaries are comprised of: a code size; a call site profile count; a critical path length; an execution time; a node level; a level criticality; and a total execution count.

18. The method of claim 16, wherein an arbitrary inlining order among the call sites in the call graph is selectable in an inline analysis phase, and wherein that same order is followed in an inline transformation phase.

19. The method of claim 16, wherein a measure of goodness for each call site is computed from the light-weight inliner summary, and a total profit is computed as a product of component profit factors, and a total cost is computed as a product of component cost factors.

20. The method of claim 16, wherein information in the light-weight inliner summary corresponding to call-graph nodes and edges are updated each time a call site is accepted for inlining, and wherein a goodness factor of each call site with updated summary information is re-computed.

21. The method of claim 16, wherein a globally-sorted work list and an associated table are maintained and used to continuously order the work list and extract a call site with a highest goodness factor.

EVIDENCE APPENDIX

There are no documents or items submitted under this section.

RELATED PROCEEDINGS APPENDIX

There are no documents or items submitted under this section.